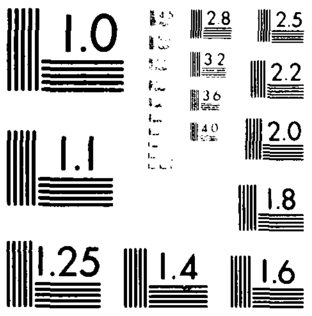


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts 02139
617 253-1000

Contract Number N00039-80-K-0573
Internal Report Number P010-8007-14
Deliverable Number 002

12

A GRAPH DECOMPOSITION TECHNIQUE BASED ON A HIGH-DENSITY CLUSTERING MODEL ON GRAPHS

M. Anthony Wong

Technical Report #14
July 1980

Principal Investigator:
Professor S.E. Madnick

Prepared for:
Naval Electronic Systems Command
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Technical Report #14 ✓	2. GOVT ACCESSION NO: AD-A090 348	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A Graph Decomposition Technique based on a High-Density Clustering Model on Graphs		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER: P010-8007-14
7. AUTHOR(s): M.A. Wong		8. CONTRACT OR GRANT NUMBER(s): N00039-80-K-0573
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center for Information Systems Research, Sloan School of Management, M.I.T., Cambridge, MA 02139 ✓		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE: Jul 80
		13. NUMBER OF PAGES: [illegible]
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report): Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software systems design; graph decomposition; high-density clustering model; minimum spanning tree.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Complex design problems are characteristized by a multitude of competing requirements. System designers frequently find the scope of the problem beyond their conceptual abilities, and attempt to cope with this difficulty by decomposing the original design problem into smaller, more manageable sub-problems. Functional requirements form a key interface between the users of a system and its designers. In this research effort, a systematic approach has been proposed for the decomposition of the overall set of functional requirements into sub-problems to form a design structure (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

that will exhibit the key characteristics of good design: strong coupling within sub-problems, and weak coupling between them.

A scan of the graph decomposition algorithms review that none of the existing techniques is particularly well suited for use in Systematic Design Methodology. In this report, a clustering model on a graph is defined, using the concept of "natural" or "high density" clusters which are densely-connected subgraphs separated by relatively few links. A graph decomposition techniques based on this "high-density" clustering model is developed for identifying the best "natural" partition for a given graph. Numerous examples are included to illustrate its effectiveness.

EXECUTIVE SUMMARY

Complex design problems are characterized by a multitude of competing requirements. System designers frequently find the scope of the problem beyond their conceptual abilities, and attempt to cope with this difficulty by decomposing the original design problem into smaller, more manageable sub-problems. Functional requirements form a key interface between the users of a system and its designers. In this research effort, a systematic approach has been proposed for the decomposition of the overall set of functional requirements into sub-problems to form a design structure that will exhibit the key characteristics of good design: strong coupling within sub-problems, and weak coupling between them.

A scan of the graph decomposition algorithms review that none of the existing techniques is particularly well suited for use in Systematic Design Methodology. *In this report, a clustering model on a graph* is defined, using the concept of "natural" or "high density" clusters which are densely-connected subgraphs separated by relatively few links. A graph decomposition techniques based on this "high-density" clustering model is developed for identifying the best "natural" partition for a given graph. Numerous examples are included to illustrate its effectiveness.

TABLE OF CONTENTS

1. Introduction

The cost of software development, testing, and maintenance in 1979 is estimated to have exceeded $40 billion. In several military computer systems, it has been found that about 80 to 90 percent of the acquisition costs go for software (Computer, p. 62, January 1979). Yet, the approaches to software requirements definition and "architectural design", where the overall design is initially divided into sub-tasks, are still largely intuitive and subjective. It has been noted that many of the problems with managing such projects emanate from mistakes and omissions made during these early decision stages. In one study of project managers (Computer, p. 67, January 1979), the major problem reported, with 100 percent agreement, was requirements specification.

The Systematic Design Methodology (SDM) research effort is directed at helping to structure the early stages of system design, providing a quantitative measure of design quality, improving the architectural design process, and supporting the subsequent steps of the project life-cycle. By "methodology" we mean: (1) a set of basic underlying concepts, (2) a set of techniques and algorithms, and (3) a set of tools for implementing the techniques (e.g., a computerized design support system).

The SDM concept is based upon four key steps: (1) formation of formal implementation - independent functional requirements, (2) assessment of interdependencies between requirements to form a graph structure, (3) decomposition of the graph into sub-graphs, and (4) interpretation of the subgraphs as design sub-problems.

STEP3, the graph decomposition step, is particularly important. In the design literature, an optimal partitioning of a project is typically defined qualitatively as one where each task is well-defined and there is minimal interaction between individual tasks. Our aim is to identify or develop a graph decomposition technique that would correspondingly partition a graph into well-connected subgraphs with minimal cross-links among them. In Section 2, the results of a literature research on graph decomposition techniques are presented. It is pointed out that the important problem: "how many separate well-connected subgraphs are there?" did not receive much attention, and that existing goodness-of-partition measures for finding the best partition tend to favor extreme solutions (either partitions with many small subgraphs or a few big subgraphs).

This leads to the development of a "high-density" clustering model on a graph, in which high-density clusters can be perceived as well-connected subgraphs separated by relatively few cross-links. The model is described in Section 3, where a decomposition technique for finding the best partition based on this clustering model is also given. Numerous examples are included to illustrate the effectiveness of this new graph decomposition technique.

2. Literature Research

The graph decomposition literature can be divided into two types: (i) Criterion - optimizing techniques - in which the clusters of nodes are formed by optimization of a goodness-of-partition criterion. (ii) Density or bond-energy or node-tearing techniques - in which clusters are formed by searching for well-connected subgraphs separated by relatively few links.

2.1. Criterion-optimizing techniques

The majority of this type of techniques can be formulated as attempts to partition the graph so as to optimize some predetermined criterion. For example, many of them attempt to minimize the sum of the weights of the cross-links between subgraphs. (See Kernighan and Lin (1970), Lukes (1974, 1975), anc Christofides and Brooker (1976) for methods that operate under some size constraints on the subgraphs, and see Ford and Fulkerson (1962) for an unconstrained approach.) Most of the methods also assume that the number of clusters (or subgraphs) has been decided a priori by the investigator, although some do allow the number to be changed during the course of the analysis (Huff, 1979). Except for Huff's (1979) work, the problem of how to choose the "correct" number of clusters (or subgraphs) was never addressed.

The differences between the methods lie primarily in the optimality of the final solution and the efficiency of the computation algorithm. Gorinshteyn (1969), Lukes (1975), and Christofides and Brooker (1976) all set out to find the globally optimal solution; however, Gorinshteyn uses a branch-and-bound strategy and Lukes take the dynamic programming approach, while Christofides and Brooker adopt the tree search method. On the other hand, Kernighan and Lin (1970) and Huff (1979) use an efficient interchange heuristic to search for a locally optimal solution.

Except for Huff's (1979) interchange algorithm, the various criterion -

optimizing techniques are not useful in the SDM context as they do not address the problem of how to choose the "correct" number of clusters. Unfortunately, the goodness-of-partition measure used by Huff has some inherent problems. His measure M is defined as follows:

$$M = \sum_{i=1}^{k} S_i - \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} C_{ij}$$

where k = number of subgraphs

S_i = strength of subgraph $i = \dfrac{L_i - (n_i - 1)}{\binom{n_i}{2} - (n_i - 1)}$ where L_i = the number of links contained within subgraph i,

n_i = the number of nodes contained within subgraph i.

and C_{ij} = *coupling between subgraphs* i *and* $j = \dfrac{L_{ij}}{n_i \cdot n_j}$ *where* L_{ij} = *the number* of links connecting nodes in subgraph i to nodes in subgraph j.

In spite of the fact that its definition has some intutitve appeal, the measure M is not a useful criterion for comparing partitions with different number of subgraphs as it favors a solution with large number of subgraphs. Specifically whenever a single link is removed from a completely-linked subgraph of any size, the value for M for the corresponding 2-subgraphs solution is greater than that for the 1-graph solution!

The problem of finding a appropriate goodness-of-partition criterion that is useful for comparing partitions with different number of subgraphs is a difficult one. However, this problem does not exist when the number of subgraphs is fixed. All criteria are equivalent to the sum of the weights of the cross-links of the subgraphs, as minimizing this sum is equivalent to maximizing the weights of the links within the subgraphs themselves. But this sum does not provide a merit index for the corresponding partition. To this end, we suggest using the following measure M^*, which is a modification of M:

$$M^* = \frac{\sum_{i=1}^{k} [L_i - (n_i-1)]}{\sum_{i=1}^{k} [\binom{n_i}{2} - (n_i-1)]} - \frac{\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} L_{ij}}{\sum_{i=1}^{k-1} \sum_{j=1+1}^{k} n_i \cdot n_j}$$

Since M^* is the difference of two weighted averages (the strengths and the couplings), its value ranges from -1 to +1 with +1 indicating a perfect partition. Hence, it is a reasonable index of merit for a given partition.

Unfortunately, M^* is itself not a useful goodness-of-partition measure as it also fails to do a good job in comparing partitions with different number of clusters. In general, it favors a large number of similar-sized, fully-connected subgraphs. However, there is some indication that it might become useful when size constraints are put on the subgraphs.

2.2 Density or bond-energy or node-tearing techniques

For a given graph, a natural concept of clustering suggests that there should be parts of the graph (i.e., subgraphs) in which the nodes are densely-connected, separated by relatively few cross-links. (For related concepts in Euclidean space, see Zahn (1971), Hartigan (1975), and Wong (1979).) As we shall see in Section 3, this concept can be used to define "natural" or "high-density" clusters. Graph decomposition methods which use this approach of seeking densely-connected subgraphs (or modal subgraphs) in the graph will be described here. In general, each modal subgraph is taken to signify a different cluster.

The first method we will consider is the bond-energy algorithm due to McCormick et al (1972). By permuting the rows and columns of the graph adjacency matrix in such a way as to push numerically larger array element together, their algorithm serves to identify the "natural groups" or "high-density clusters" in the graph, as well as the associations of these groups with one another. Thus, the bond-energy algorithm can be viewed as a density method even though it tends to give excessive weights to large elements in the array. Its main drawback is that the best partition is not explicitly given and personal judgment has to be exercised to identify the subgraphs. Moreover, its computational time increases with N^3, where N is the number of nodes.

The central idea underlying the node-tearing techniques is to locate the smallest separating sets - that is, the set of nodes with the lowest cardinality such that their removal from the graph splits the graph into two or more unconnected subgraphs. In effect, this type of technique searches for nodes which are not densely connected with other nodes, and hence their removal will reveal densely-connected subgraphs. Hence, the node-tearing techniques are also density-related methods; but their aim is to locate "loose" nodes (or low-density nodes) rather than the densely connected modal subgraphs. An efficient heuristic algorithm for node tearing was developed by Sangiovanni-Vincentelli et al (1977) in the context of electrical circuit design; the computational expense is bounded by $O(Nb)$, where b is the number of links or edges in the graph. The fact that the solution produced by this algorithm is only locally optimal is a major concern, in spite of the fact that they have incorporated quite a few intuitively sound strategies in improving their algorithm. A minor deficiency of this algorithm is that it can only be applied to an unweighted graph.

Andreu (1978, pp. 124-133) derived an algorithm for identifying leader subgraphs, subgraphs that are non-overlapping and strongly coherent. The partitioning procedure is then completed by assigning the left-over nodes (if any) to the leaders. This approach is intuitively appealing and is directly related to the "high-density" concept (see Section 3). However, his effort fell short of describing the high-density clustering model on a graph, and hence he has problems defining the assighment rules for the leftover nodes.

It is clear that the density techniques are appropriate for use in the SDM context. However, none of the existing techniques appear to be completely satisfactory. In the next section, the high-density clustering model on a graph will be described. Moreover, it will be shown how the optimal partition into "natural" or "high-density" subgraphs can be derived from the tree of high-density clusters on a graph. Another feature of the partitioning algorithm related to this model is its flexibility in allowing the user some control over the size of the resulting subgraphs or the number of subgraphs.

3. High Density Clusters on a Graph

3.1. The High-density Clustering Models

For a given graph, clusters of nodes may be thought of as densely-connected subgraphs separated from other such subgraphs by relatively few cross-links. To formalize this concept we need to first define the density contour on the link between any two nodes i and j ; the level of the contour is uded to signify the extent of linkage or connectedness between the two nodes. Intuitively, two nodes should not belong to the same cluster simply because they are linked (see Figure 1),

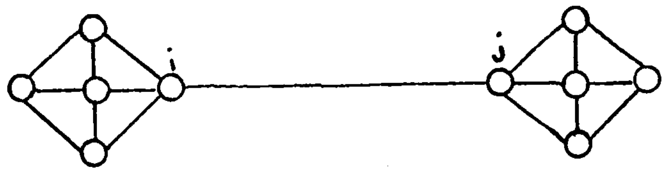


Figure 1
Nodes i and j should not be in the same cluster.

nor should they be in the same cluster simply because they are both linked to many different nodes (see Figure 2).

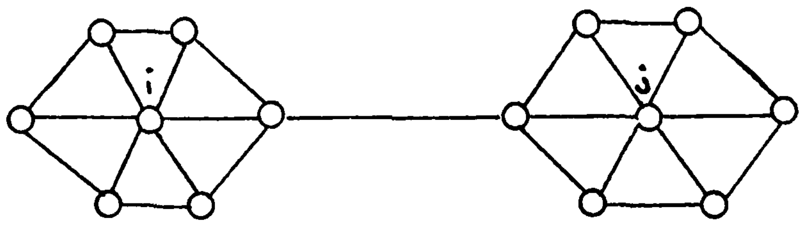


Figure 2
Nodes i and j should not be in the same cluster.

However, they do belong to the same cluster if they are linked to each other and are linked to many common nodes (see Figure 3).

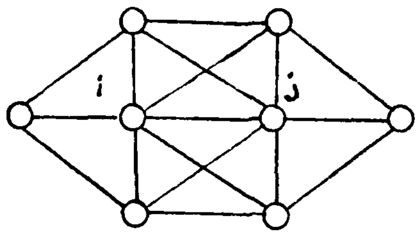


Figure 3
Nodes i and j should be in the same cluster

Using this neighborhood concept (which corresponds to the nearest neighbor technique in statistical density estimation), the density contour on the link between two linked nodes i and j is defined as follows:

$$d_{ij} = \frac{\cap N_{ij}}{\cup N_{ij}} = \frac{\text{no. of nodes connected to both } i \text{ and } j \text{ (} i \text{ and } j \text{ included)}}{\text{no. of nodes connected to either } i \text{ or } j \text{ or both (} i \text{ \& } j \text{ included)}}$$

(The contour between two un-linked nodes is defined to be zero.) Using this definition, the density contour between nodes i and j in Figure 1, 2, and 3 are respectively 2/8, 0, and 6/8. Moreover, the contour on each of the links in Figure 1 has been computed and are given in Figure 4 to illustrate the calculation.

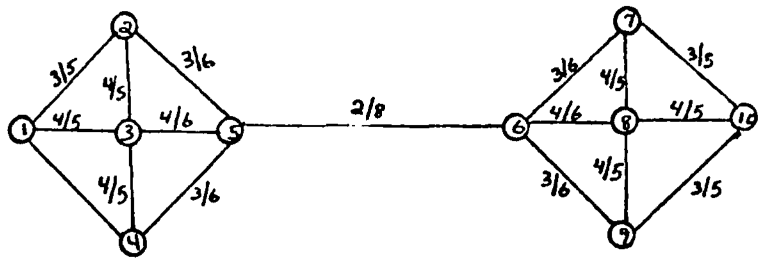


Figure 4
Density Contours on an Unweighted Graph

For weighted graphs, the corresponding definition is

$$d_{ij} = \frac{2W_{ij} + \frac{1}{2} \sum_{\kappa \in C} (W_{ik} + W_{jk})}{\cup N_{ij}}$$

where W_{ij} = weight on the link between nodes i and j and C is the set of nodes connected to both i and j (i and j excluded). Now, we are ready to define the high-density clusters on a graph: Definition: A high-density cluster at level d* on a graph G is a subgraph S such that S is maximal among connected sets of nodes whose nodes are connected by links with density contour ≥ d*.

It is easy to show that the family of high-density clusters on a graph forms a tree. An example of high-density clusters is given in Figure 5; the density contours on the graph are taken from Figure 4.

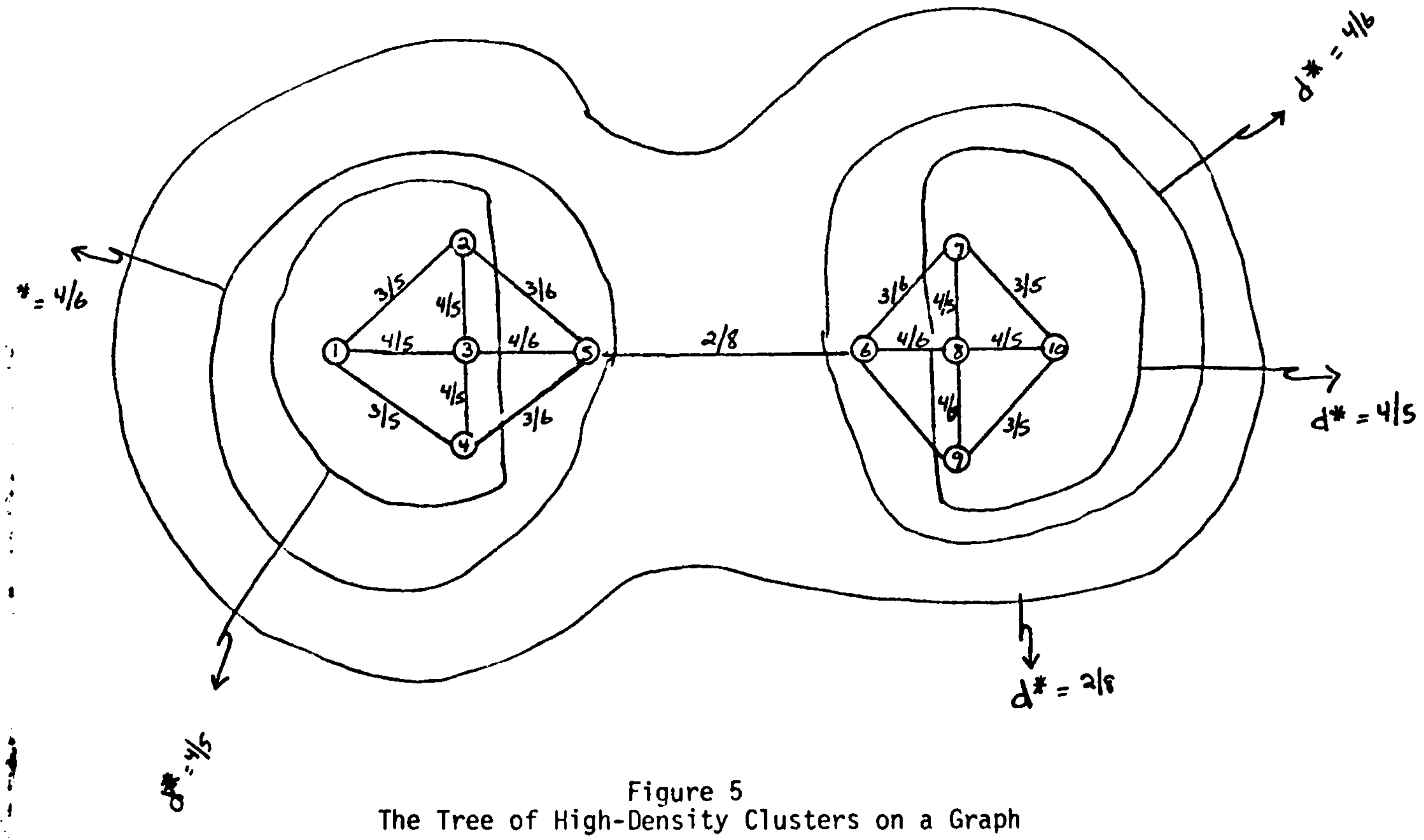


Figure 5
The Tree of High-Density Clusters on a Graph

As the density level d^* is reduced, the subgraph S at level d^* gradually expands until suddenly, at some splitting level d^*_s, it coalesces with other subgraphs to form a much larger subgraph. These branching clusters which cannot be expanded smoothly, are important in determining the tree structure. They also signify the number of (high-density) modal subgraphs in a given graph. For example, if there is only one such cluster, there is only one modal subgraph and it would be artificial to partition the graph in any way. A high-density cluster S is a branching cluster if every cluster properly including it contains a cluster disjoint from S. The branching high-density clusters form a tree that is a subset of the tree of all high-density cluster.

3.2. The Minimum Spanning Tree Algorithm for Computing the Tree of High-Density Clusters

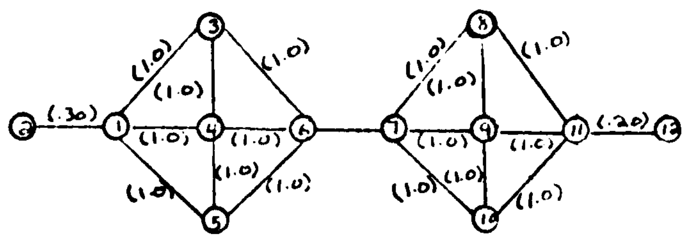
Given a set of N nodes with density contour $d(i,j)$, the algorithm for finding the tree of high-density clusters is as follows:

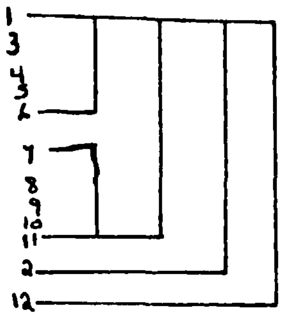
STEP1. Let i and j be the pair of nodes with the densest link, amalgamate them to form a cluster l and define the density contour between that cluster and any node k by

$$d(l,k) = \max\ \{d(i,k), d(j,k)\}$$

STEP2. Repeat STEP1 treating l as a node and ignoring i and j. The amalgamation continues until all nodes are grouped into one large cluster. (All clusters obtained in the course of the algorithm are high density clusters.) It is easy to see that this algorithm is equivalent to the classical minimum-spanning tree (MST) algorithm. For published computational algorithm that would produce the MST, see for example, Ross (1969) and Hartigan (1975).

Unfortunately, the standard tree-like clustering output is often misleading for partitioning purposes as it fails to report the neighborhood of nodes that are not in any branching cluster. For example, in Figure 6, the information concerning node 2's connection to the branching subgraph, {1, 3, 4, 5, 6} is completely lost in the tree representation.





Graph (density contours are shown)

Tree output

Figure 6
Problem of Defining a Partition from the Tree output

Hence, we need an algorithm that would extract the correct partition from the tree, perhaps by assigning "outlying" or "leftover" nodes to branching high-density clusters using the notions of connectedness and level of density contour.

3.3. Extracting Partitions from the Tree of High Density Clusters

To extract the correct partitions from the tree of high density clusters, we need to do the following:

STEP1. Determine the number of minimal branching clusters, call it k. (A branching cluster is minimal if it does not properly contain another branching cluster).

STEP2. Identify the k minimal branching high-density clusters as leader subgraphs.

STEP3. Assign each of the leftover nodes according to the following rules:

-level A: consider leftover node i, let node j be the one with the highest contour level between it and node i, then node i and node j will be in the same cluster;

-level B: (if ties exist at level A): consider leftover node i, find the sum of the density on its links to various candidate branching clusters (if a tie-node is a leftover node itself, this sum is defined to be zero), and assign node i to the one with the largest sum;

-level C: (if ties exist at level B): random assignment among the candidate branching clusters or among the other leftover nodes.

3.4. Illustrative Examples

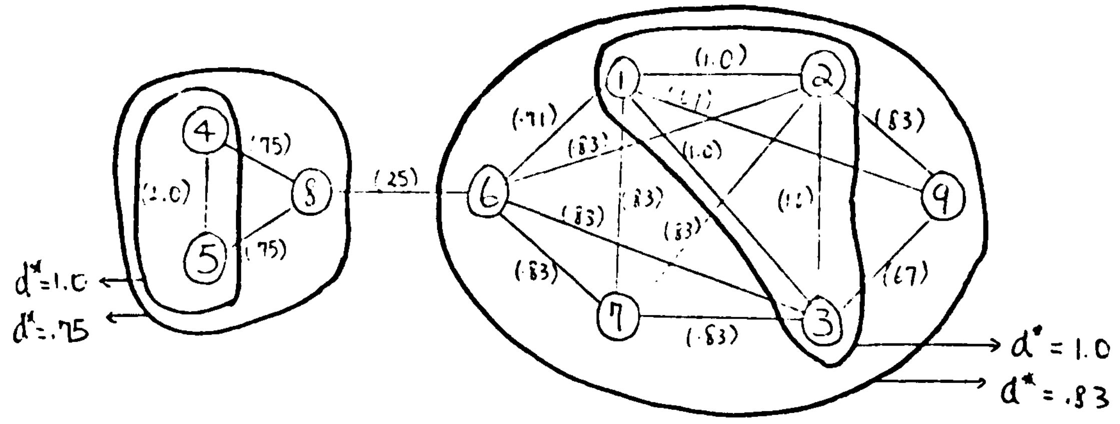
Nineteen examples are included in this section to illustrate the effectiveness of the high-density clustering approach. In Examples 1, 2, and 3, the complete tree of high-density clusters is given together with the density contour computed for the various links in the graph; but, in all subsequent examples, only the tree of branching high-density clusters is shown. In all cases, the partition obtained by the algorithm described in Section 3.3 are shown; in some, alternative solutions are also given to illustrate the flexibility (though admittedly limited) of our method. The measure M*, described in Section 2.2, will be used to indicate the merit or goodness of each decomposition.

Example 1

Number of nodes = 9

Number of links = 18

Note 1: the density contours are in parentheses



Two Branching Clusters; best partition is:

Subgraph 1: {4, 5, 8}

Subgraph 2: {1, 2, 3, 6, 7, 9}

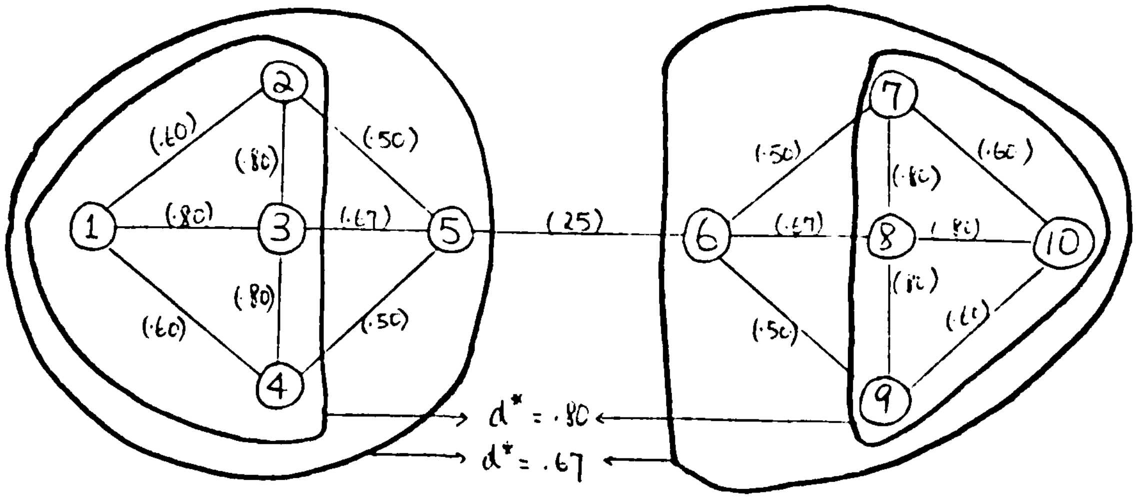
Merit Index of Decomposition: M* = .7626„

Example 2

Number of nodes = 10

Number of links = 17

Note 1: The density contours are in parentheses.



Two Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 4, 5}

Subgraph 2: {6, 7, 8, 9, 10}

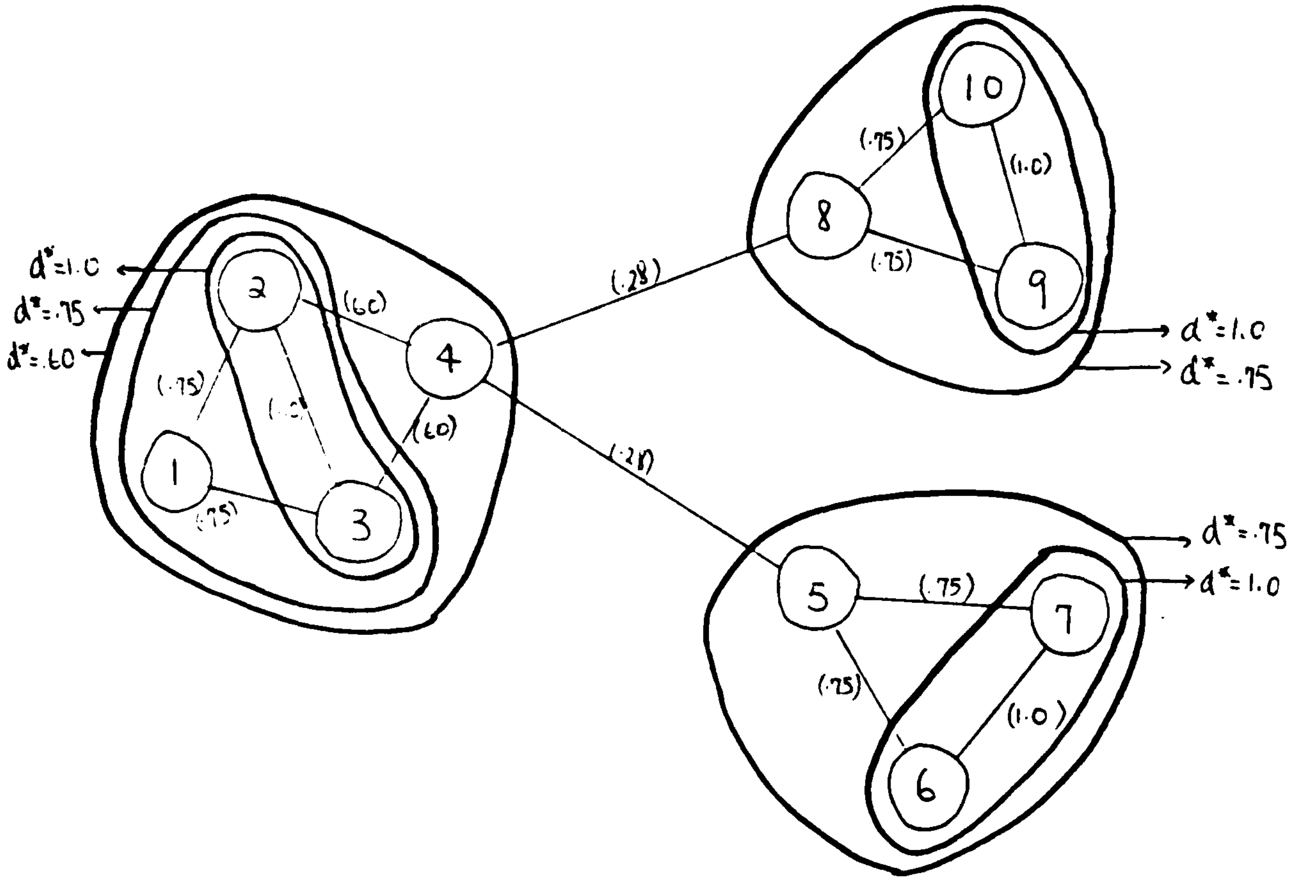
Merit Index of Decomposition: M* = .6267.

Example 3

Number of nodes = 10

Number of links = 13

Note 1: The density contours are in parentheses.



Three Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 4}

Subgraph 2: {5, 6, 7}

Subgraph 3: {8, 9, 10}

Merit Index of Decomposition: M* = 7394.

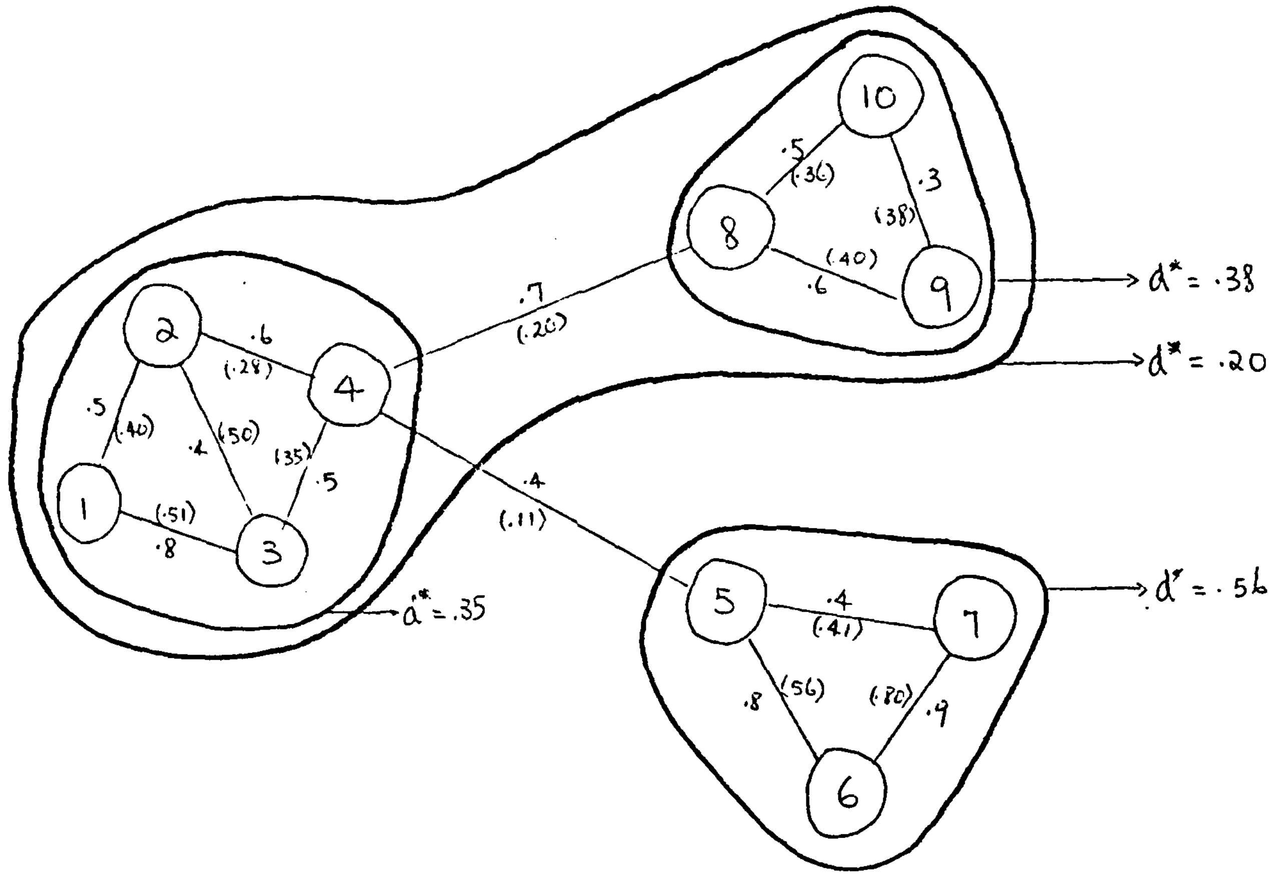
Example 4

Number of nodes = 10

Number of links = 13

Note 1: The density contours are in parentheses.

Note 2: This is a weighted graph. (average weight per link = .57)



Three Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 4}

Subgraph 2: {8, 9, 10}

Subgraph 3: {5, 6, 7}

with M* = .4240.

Alternative Solution (best partition for two subgraphs)

Subgraph 1: {1, 2, 3, 4, 8, 9, 10}

Subgraph 2: {5, 6, 7}

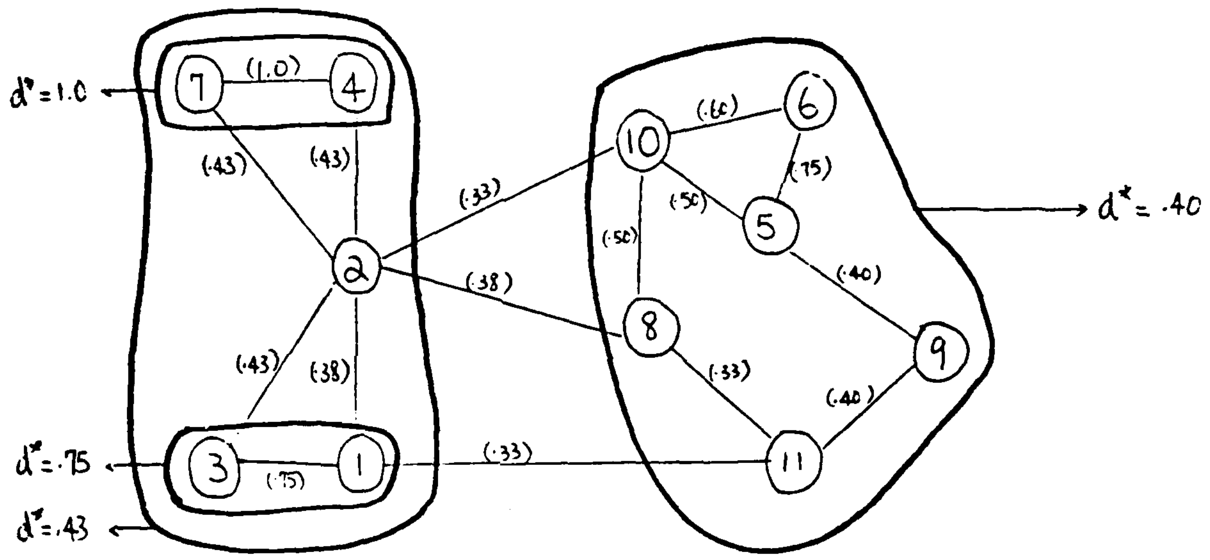
with M* = .1236„

Example 5

Number of nodes = 11

Number of links = 16

Note 1: The density contours are in parentheses.



Three Branching Clusters; best partition is:

Subgraph 1: {2*, 4, 7} *-assigned by rule at level B.

Subgraph 2: {2, 3}

Subgraph 3: {5, 6, 8, 9, 10, 11}

with M* = .1338„

Alternative Solution (best partition for two subgraphs)

Subgraph 1: {1, 2, 3, 4, 7}

Subgraph 2: {5, 6, 8, 9, 10, 11}

with M* = .1500„

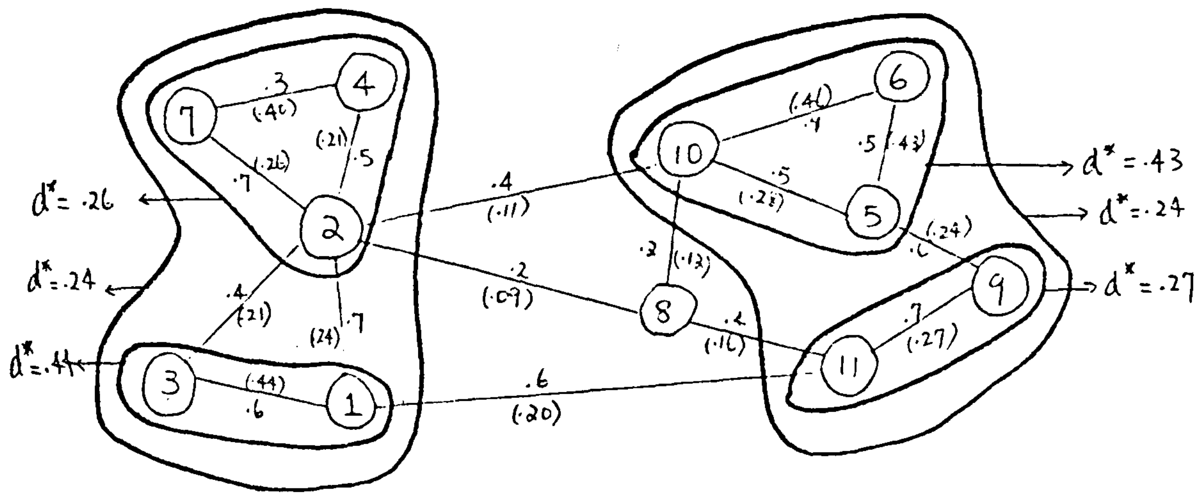
Example 6

Number of nodes = 11

Number of linsk = 16

Note 1: The density contours are in parentheses.

Note 2: This is a weighted graph. (average weight per link = .51)



Four Branching Clusters; best partition is:

Subgraph 1: 2, 4, 7

Subgraph 2: 1, 3

Subgraph 3: 5, 6, 10

Subgraph 4: 8*, 9, 11 *-assigned by rule at level A.

with M* = .3089"

Alternative Solutions (best partition for two and three subgraphs)

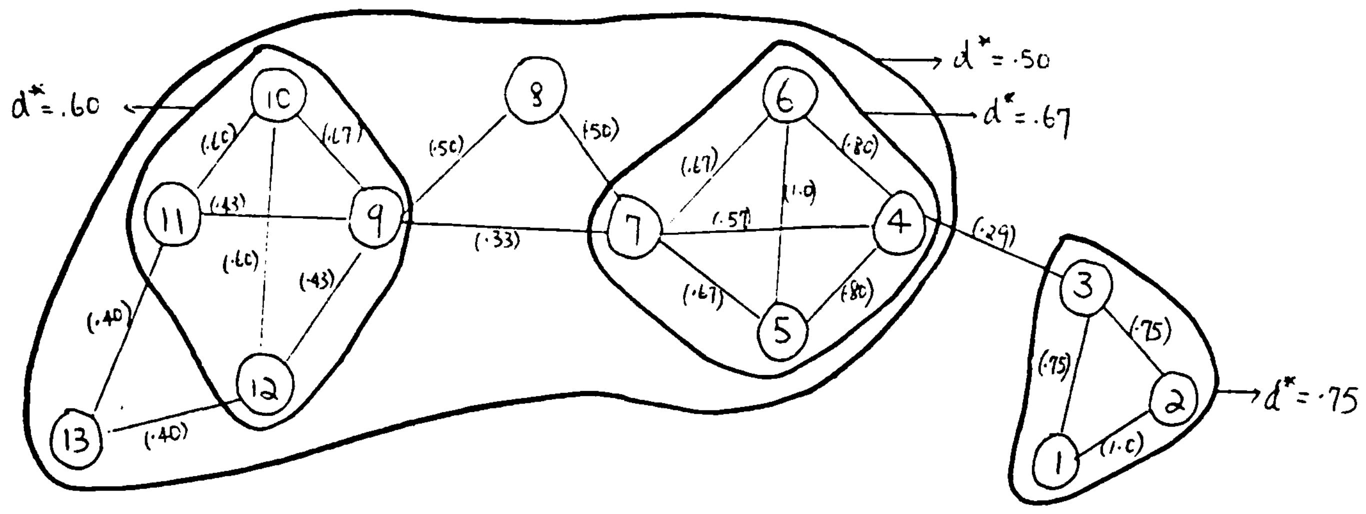
Subgraph 1: {1, 2, 3, 4, 7}	Subgraph 1: {1, 2, 3, 4, 7}
Subgraph 2: {5, 6, 8*, 9, 10, 11}	Subgraph 2: {5, 6, 10}
with M* = .0945"	Subgraph 3: {8*, 9, 11}
	with M* = .1612"

Example 7

Number of nodes = 13

Number of links = 20

Note 1: The density contours are in parentheses.



Three Branching Clusters; best partition is:

Subgraph 1: {9, 10, 11, 12, 13*} *-assigned by rule at level A

Subgraph 2: {4, 5, 6, 7, 8**} **-assigned by rule at level C

Subgraph 3: {1, 2, 3}

with M* = .4840.

Alternative Solution (best partition for two subgraphs):

Subgraph 1: {4, 5, 6, 7, 8, 9, 10, 11, 12, 13}

Subgraph 2: {1, 2, 3}

with M* = .1829.

Alternative Solution (best partition with size constraint: size > 3):

Subgraph 1: {1, 2, 3, 4, 5, 6, 7, 8**}

Subgraph 2: {9, 10, 11, 12, 13}

with M* = .2093.

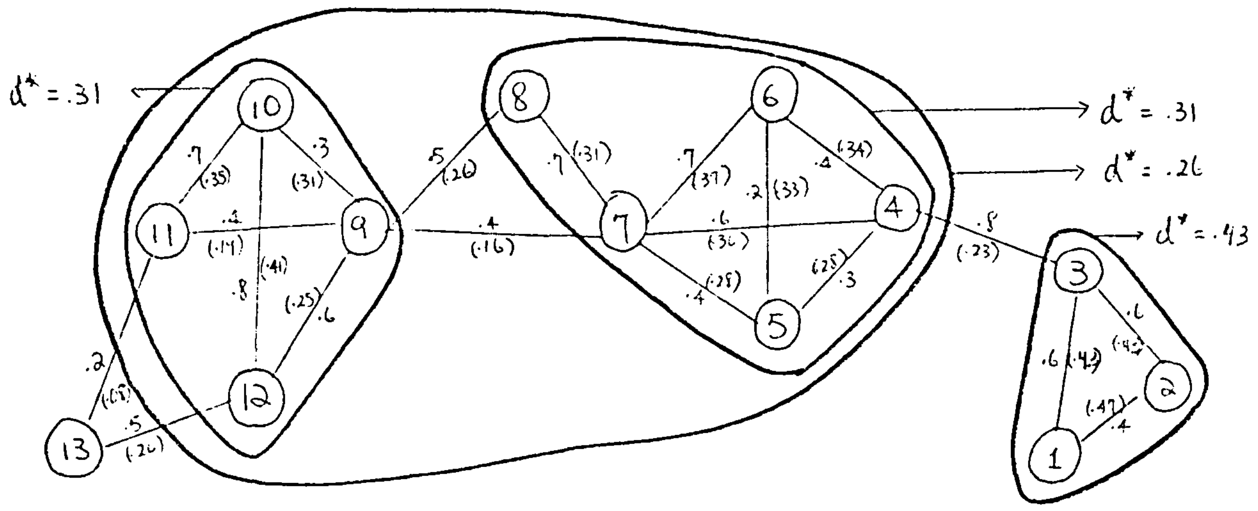
Example 8

Number of nodes = 13

Number of links = 20

Note 1: The density contours are in parentheses

Note 2: This is a weighted graph (average weight per link = .505)



Three Branching Clusters; best partition is:

Subgraph 1: {9, 10, 11, 12, 13*} *-assigned by rule at level A

Subgraph 2: {4, 5, 6, 7, 8}

Subgraph 3: {1, 2, 3}

with M* = .2343

Alternative Solution (best partition for two subgraphs):

Subgraph 1: {4, 5, 6, 7, 8, 9, 10, 11, 12, 13*}

Subgraph 2: {1, 2, 3}

with M* = .0788

Alternative Solution (best partition with constraint = size > 3):

Subgraph 1: {1, 2, 3, 4, 5, 6, 7, 8}

Subgraph 2: {9, 10, 11, 12, 13}

with M* = .1323

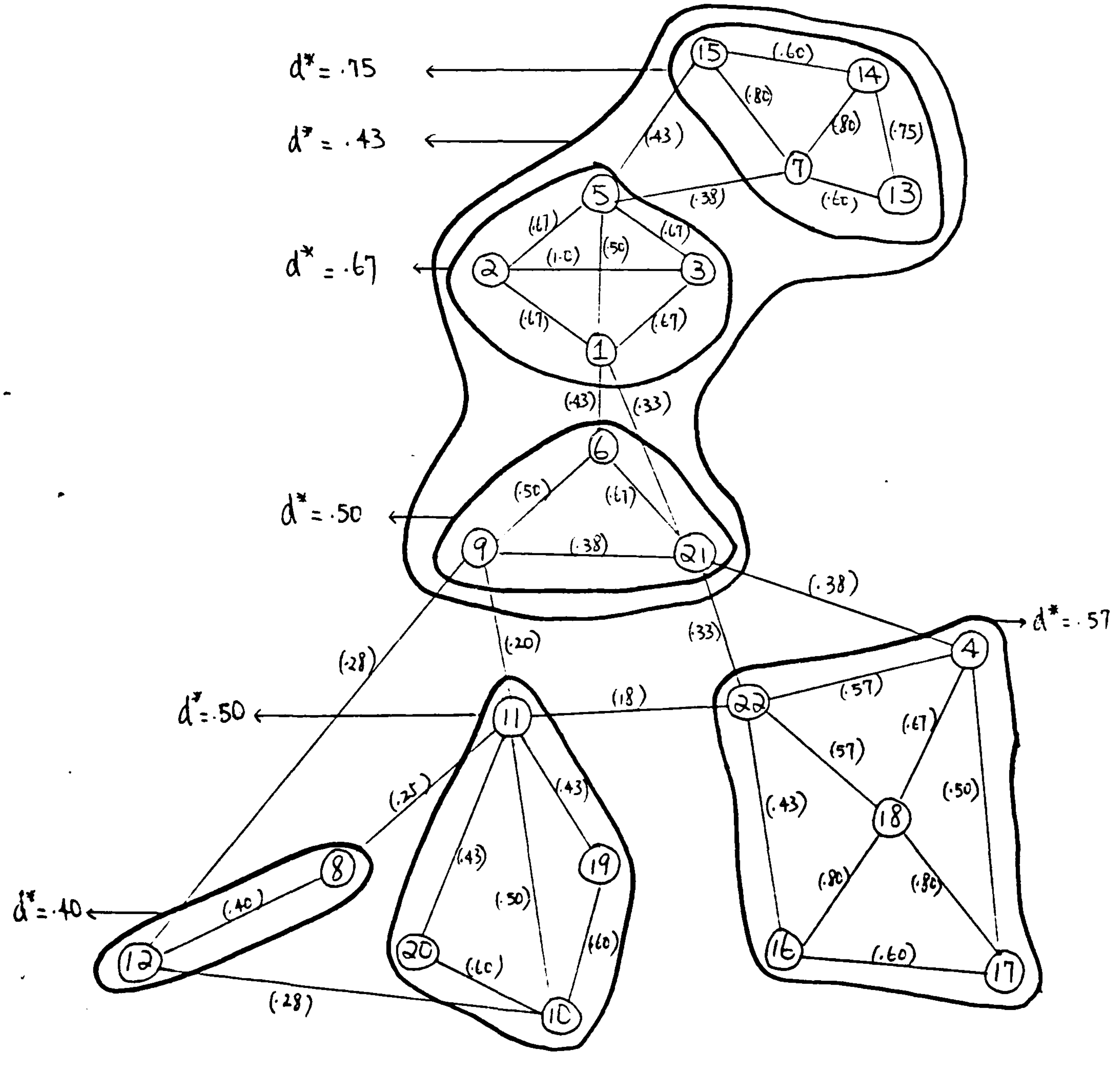
Example 9

Number of nodes = 22

Number of links = 39

Note 1: The density contours are in parentheses

Note 2: This real example is taken from a design problem in Andreu (1978).



Six Branching Clusters; best partition is:

Subgraph 1: {7, 13, 14, 15}

Subgraph 2: {1, 2, 3, 5}

Subgraph 3: {6, 9, 21}

Subgraph 4: {4, 16, 17, 18, 22}

Subgraph 5: {10, 11, 19, 20}

Subgraph 6: {8, 12}

with M* = .6947.

Alternative Solution (best partition with constraint = size > 2; also the best partition for five subgraphs):

Subgraph 1: {7, 13, 14, 15}

Subgraph 2: {1, 2, 3, 5}

Subgraph 3: {6, 9, 21}

Subgraph 4: {4, 16, 17, 18, 22}

Subgraph 5: {8, 10, 11, 12, 19, 20}

with M* = .6272.

Example 10

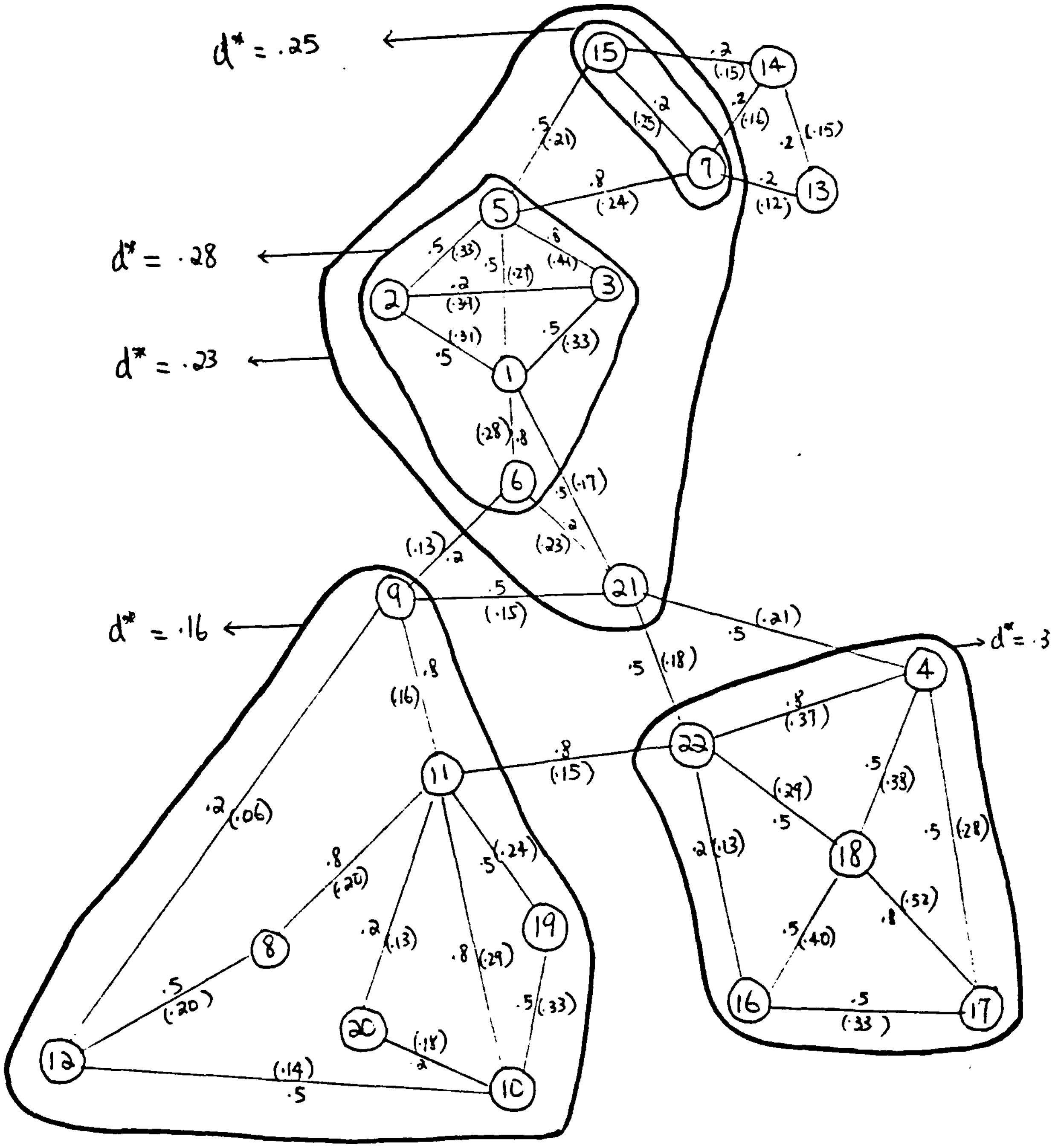
Number of nodes = 22

Number of links = 39

Note 1: The density contours are in parentheses

Note 2: This is a weighted graph. (average weight per link = .4769)

Note 3: This real example is taken from a design problem in Huff (1979).



d* = .25
d* = .28
d* = .23
d* = .16
d* = .3

Example 10

Four Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 5, 6, 21*} *-assigned by rule at level A

Subgraph 2: {7, 13*, 14*, 15}

Subgraph 3: {4, 16, 17, 18, 22}

Subgraph 4: {8, 9, 10, 11, 12, 19, 20}

with M* = .1848.

Alternative Solution (best three subgraph partition):

Subgraph 1: {1, 2, 3, 5, 6, 7, 13*, 14*, 15, 21}

Subgraph 2: {4, 16, 17, 18, 22}

Subgraph 3: {8, 9, 10, 11, 12, 19, 20}

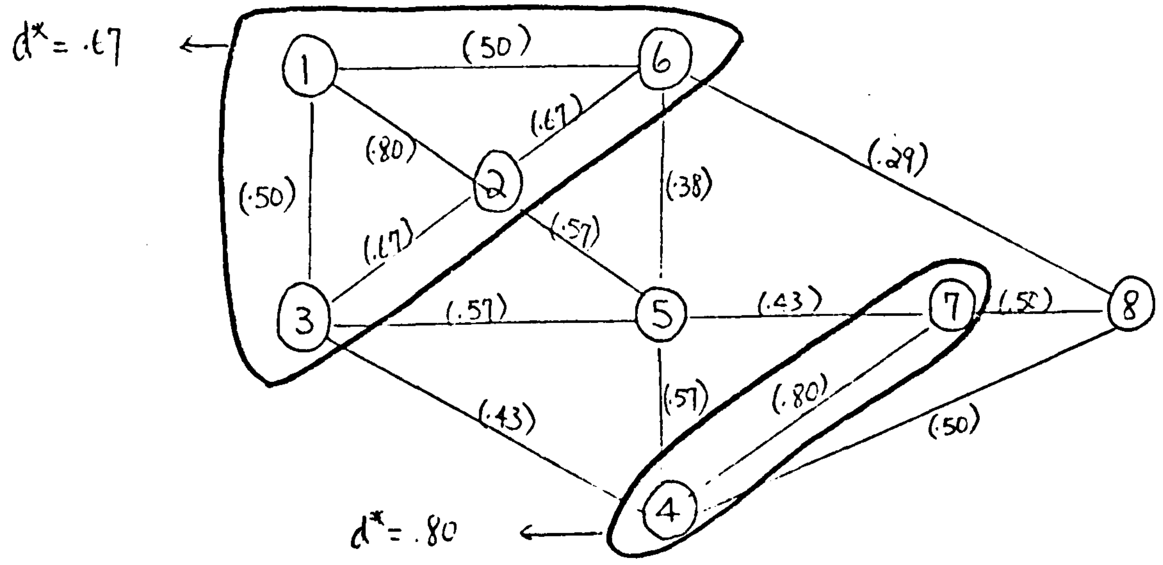
with M* = .0949.

Example 11

Number of nodes = 8

Number of links = 15

Note 1: The density contours are in parentheses



Two Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 5*, 6} *-assigned by rule at level B

Subgraph 2: {4, 7, 8**} **-assigned by rule at level A

with M* = .4476„

Example 12

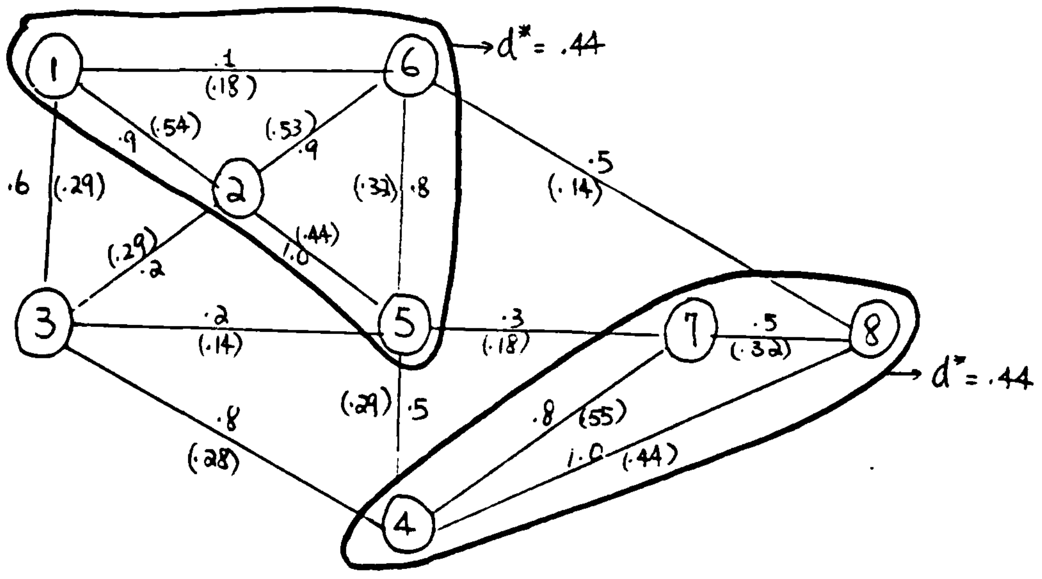
Number of nodes = 8

Number of links = 15

Note 1: The density contours are in parentheses

Note 2: This is a weighted graph. (average weight per link = .6067)

Note 3: This example is taken from Christofides and Brooker (1976).



Two Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3*, 5, 6} *-assigned by rule at level A

Subgraph 2: {4, 7, 8}

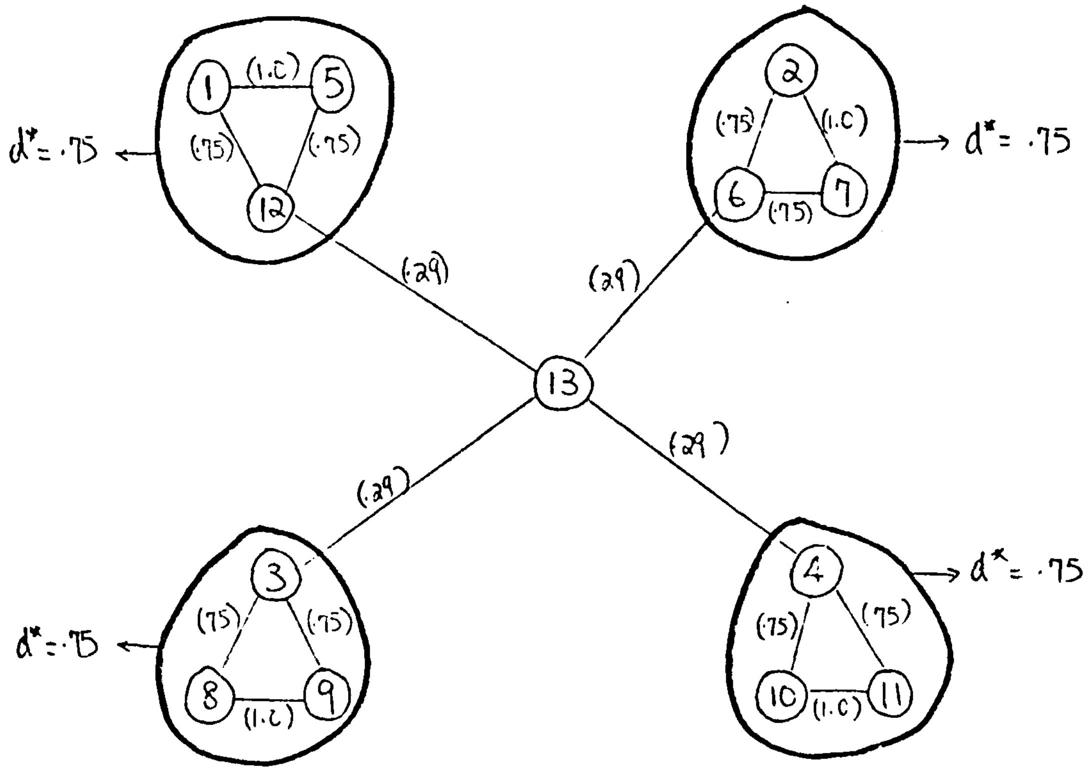
with M* = .3052„

Example 13

Number of nodes = 13

Number of links = 16

Note 1: The density contours are in parentheses



Four Branching Clusters; best partition is:

Subgraph 1: {1, 5, 12}

Subgraph 2: {2, 6, 7}

Subgraph 3: {3, 8, 9}

Subgraph 4: {4, 10, 11, 13*} *-assigned by rule at level C

with M* = .6190.

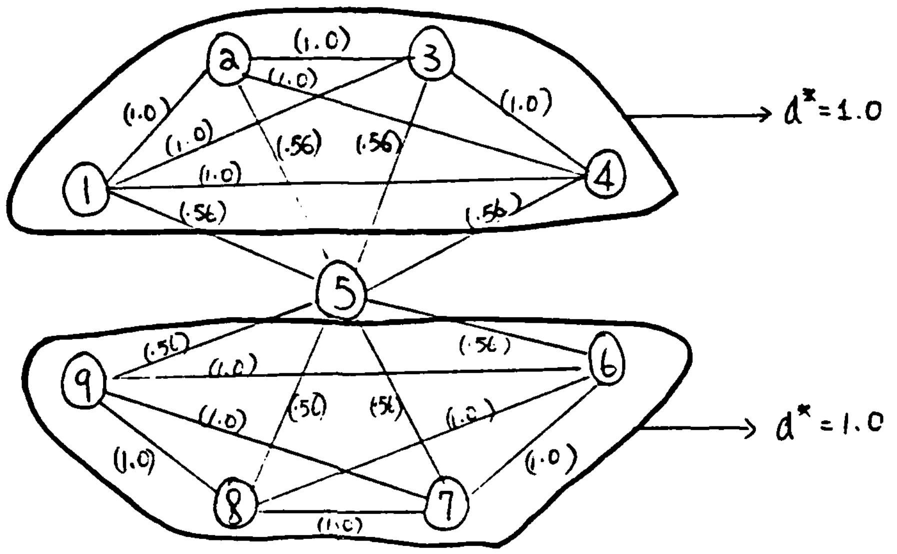
Example 14

Number of nodes = 9

Number of links = 20

Note 1: The density contours are in parentheses.

Note 2: This example is taken from Sangiovanni-Vincentelli et. al. (1977).



Two Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 4, 5*}   *-assigned by rule at level C

Subgraph 2: {6, 7, 8, 9}

with M* = .8000

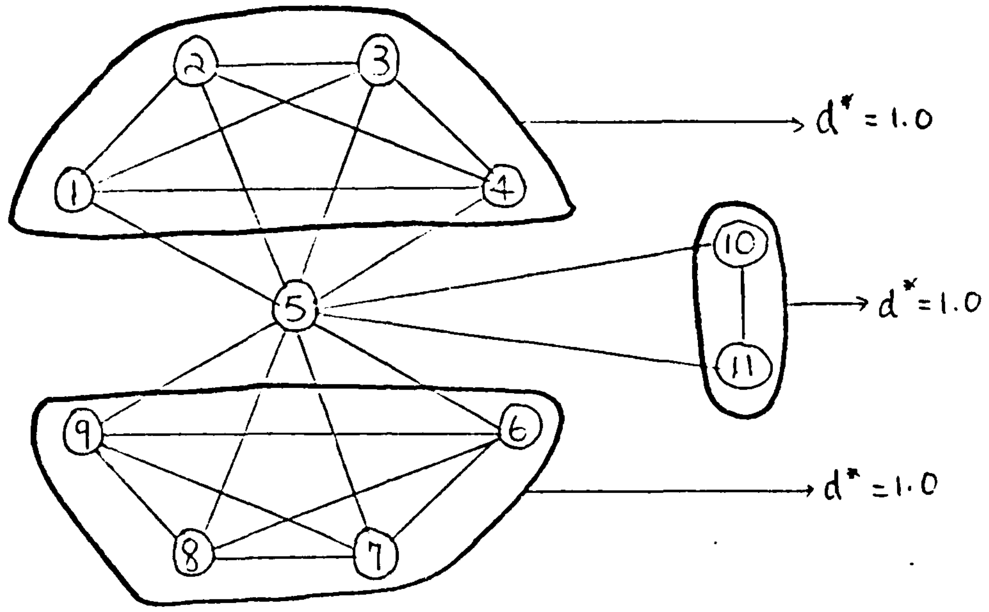
Example 15

Number of nodes = 11

Number of links = 23

Note 1: The density contours are in parentheses

Note 2: This example is taken from Sangiovanni-Vincentelli et. al. (1977).



Three Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 4, 5*} *-assigned by rule at level C

Subgraph 2: {6, 7, 8, 9}

Subgraph 3: {10, 11}

with M* = .8421„

Alternative Solution (best two-subgraph partition):

Subgraph 1: {1, 2, 3, 4, 5, 10, 11}

Subgraph 2: {6, 7, 8, 9}

with M* = .4128„

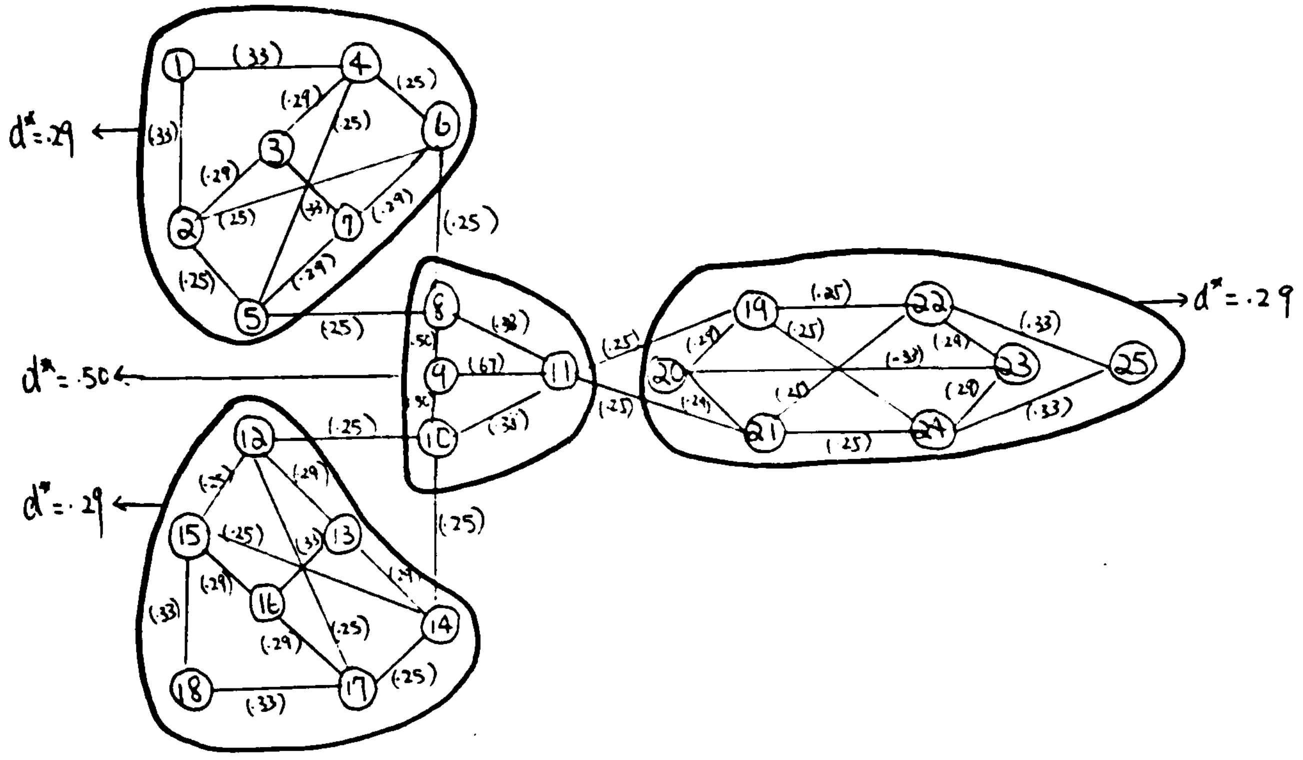
Example 16

Number of nodes = 25

Number of links = 44

Note 1: The density contours are in parentheses

Note 2: This example is taken from Sangiovanni-Vincentelli et. al. (1977).



Four Branching Clusters; best partition is:

Subgraph 1: {1, 2, 3, 4, 5, 6, 7}

Subgraph 2: {8, 9, 10, 11}

Subgraph 3: {12, 13, 14, 15, 16, 17, 18}

Subgraph 4: {19, 20, 21, 22, 23, 24, 25}

with M* = .3281.

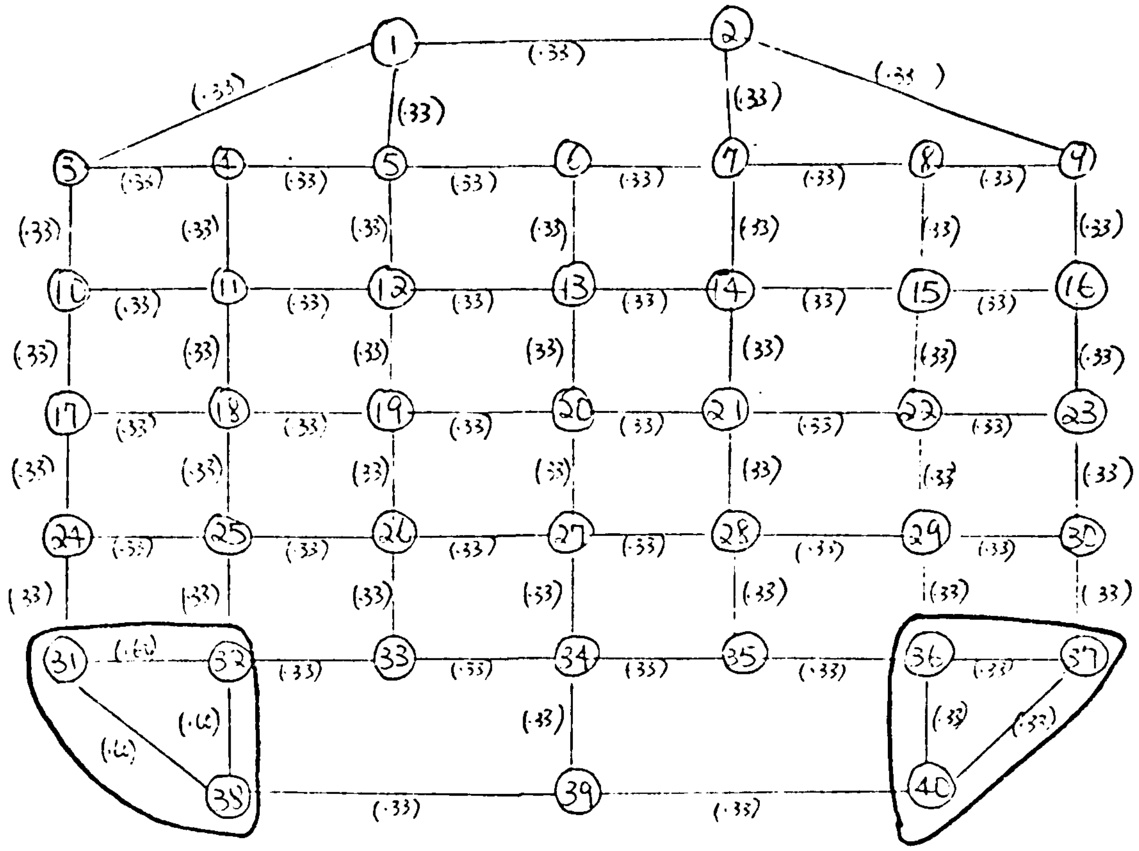
Example 17

Number of nodes = 40

Number of links = 70

Note 1: The density contours are in parenthesis

Note 2: This is an extreme example in that there is really no major high-density clusters.



Two small minor branching high-density clusters; the assignments of the remaining nodes are completely arbitrary.

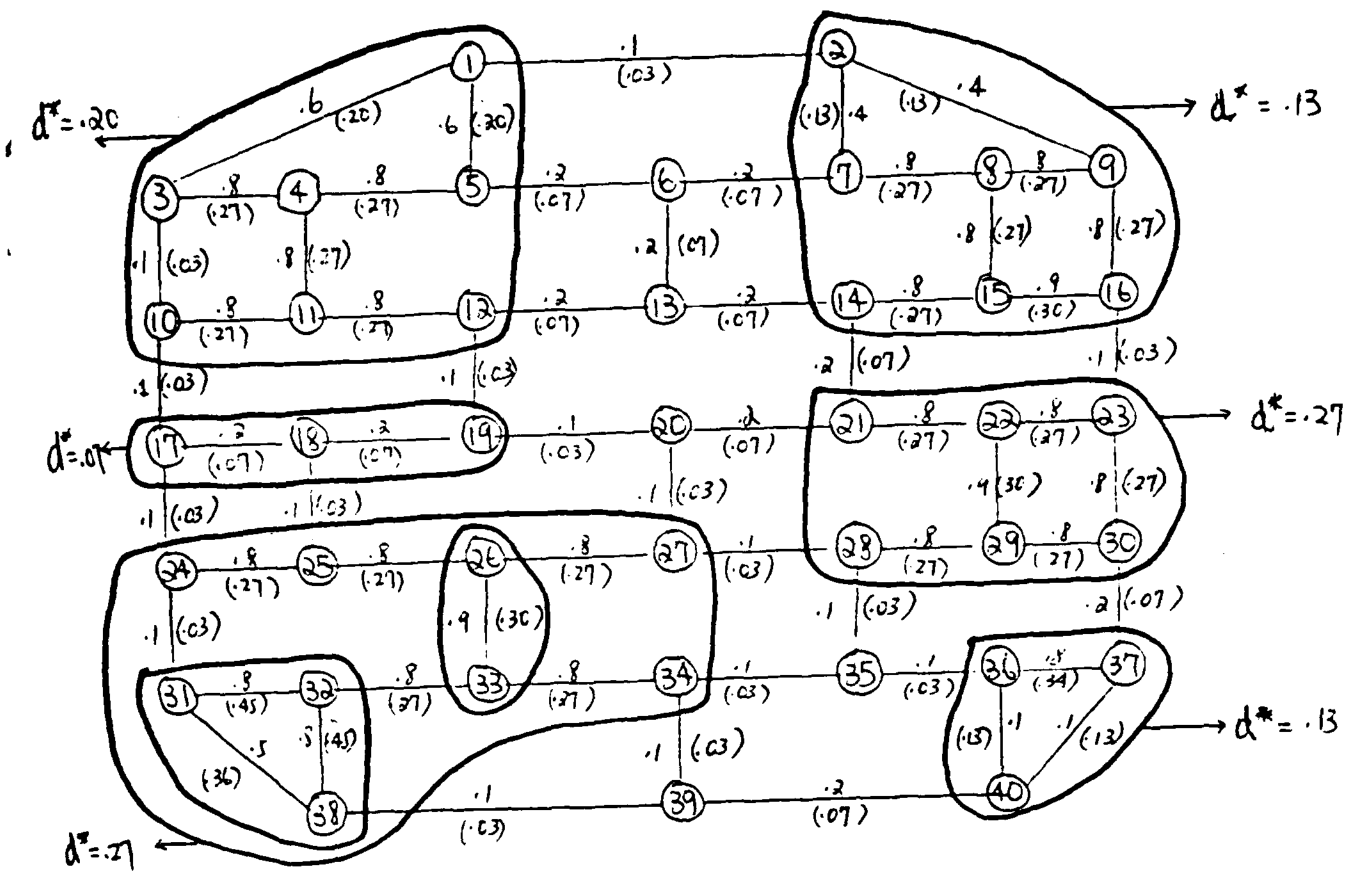
Example 18

Number of nodes = 40

Number of links = 60

Note 1: The density contours are in parentheses

Note 2: This is a weighted graph. (average weight per graph = .46)



Seven Branching Clusters; best partition is:

Subgraph 1: {1, 3, 4, 5, 10, 11, 12, 13*} *-assigned by rule at level C

Subgraph 2: {2, 6*, 7, 8, 9, 14, 15, 16}

Subgraph 3: {17, 18, 19}

Subgraph 4: {20**, 21, 22, 23, 28, 29, 30} **-assigned by rule at level A

Subgraph 6: {24**, 25**, 26, 27**, 33, 34**}

Subgraph 7: {31, 32, 28}

with M* = .0529.

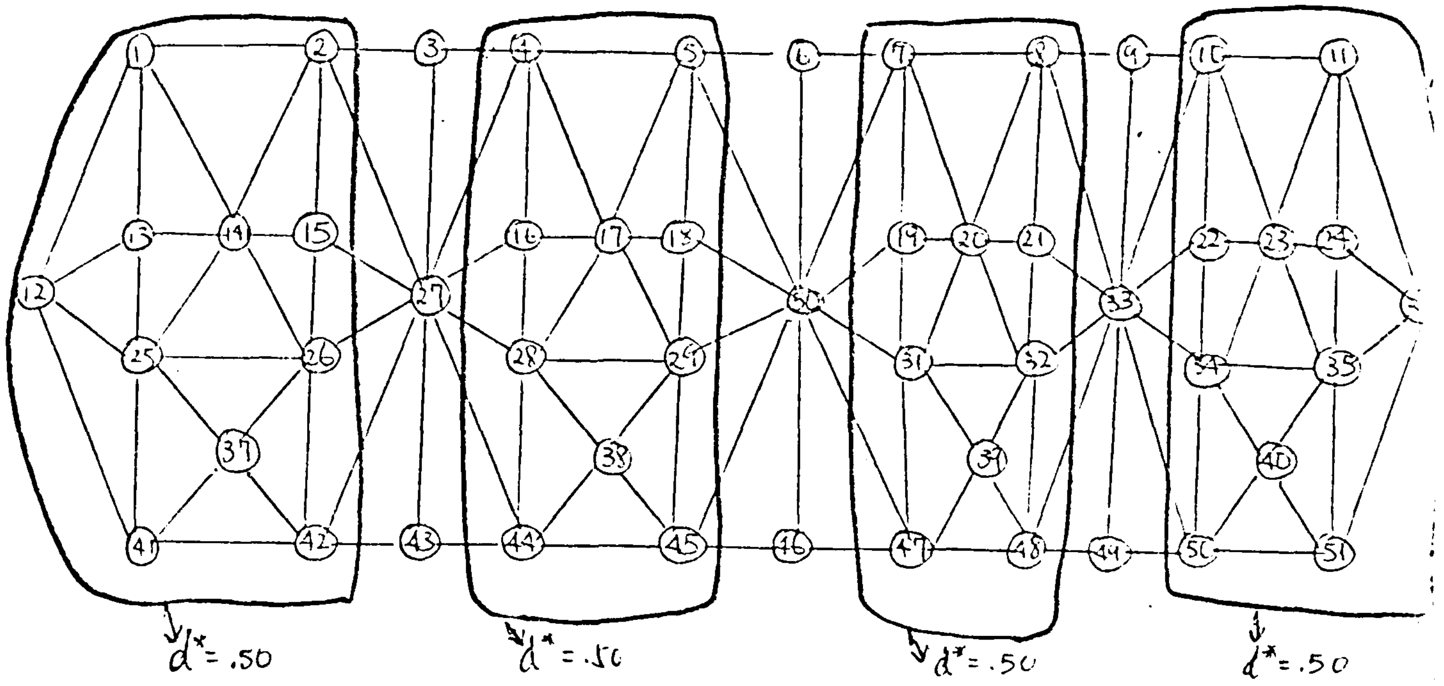
Example 19

Number of nodes = 51

Number of links = 126

Note 1: The density contours are not included here

Note 2: This example is taken from Sangiovanni-Vincentelli et. al. (1977)



Four Branching Clusters; best partition is:

Subgraph 1: {1, 2, 13, 14, 15, 25, 26, 37, 41, 42}

Subgraph 2: {3*, 4, 5, 16, 17, 18, 27*, 28, 29, 38, 43*, 44, 45} *-assigned by rule at level C

Subgraph 3: {6*, 7, 8, 19, 20, 21, 30*, 31, 32, 39, 46*, 47, 48}

Subgraph 4: {9*, 10, 11, 22, 23, 24, 33*, 34, 35, 36, 40, 49*, 50, 51}

with M* = .2207.

4. Discussion

A new algorithm for partitioning a graph into strongly coherent subgraphs, which are separated from one another by relatively few links, has been developed. The algorithm functions by extracting the appropriate partition from the tree of high-density clusters, a concept which is formalized in this report. The numerous examples included serve to illustrate the effectiveness of this decomposition technique. We believe that the high-density clustering concept is a useful contribution to the general graph decomposition methodology, as other existing techniques are directly or indirectly related to it; its relationships with other density-related techniques are pointed out in Section 2.2, and it is not difficult to see that finding the partition with the minimum sum of weights of the cross-links is equivalent to cutting the graph at some "low-density" links.

In later studies, we intend to examine the computational efficiency (which is known to be bounded by $O(N^2)$ computations) of this new approach and to explore its utility and effectiveness for real design problems.

REFERENCES

1. Andreu, R. (1978). A Systematic Approach to the Design and Structure of the Complex Software Systems. Ph.D. Dissertation, Sloan School of Management, M.I.T.

2. Christofides, N. and Brooker, P. (1976). The Optimal Partitioning of Graphs. SIAM Journal of Appl. Math., 30, 55-69.

3. Ford, L.R. and Fulkerson, D.R. (1962). Flows in Networks, Princeton University Press.

4. Forinshteyn, L.L. (1969). Partitioning of Graphs, Engrg Cybermetics, 76-82.

5. Hartigan, J.A. (1975) Clustering Algorithms, New York: John Wiley.

6. Huff, S.L. (1979). Decomposition of weighted graphs using the interchange partitioning algorithm. Technical Report #8, Center for Information Systems Research, Sloan School of Management, M.I.T.

7. Kernighan, B.W. and Lin, S. (1970). An efficient heuristic for partitioning graphs, Bell System Technical Journal, 49, 291-307.

8. Lukes, J.A. (1974). Efficient algorithm for the partitioning of trees. IBM J. of Research and Development, 18, 217-224.

9. Lukes, J.A. (1975). Combinatorial solutions to the partitioning of general graphs. IBM J. of Research and Development, 19, 170-190.

10. McCormick, W.T., Schweitzer, P.J., and White, T.W. (1972). Problem Decomposition and Data Reorganization by a Clustering Technique. Operations Research, Vol. 20, No. 5, 993-1007.

11. Ross, G.J.S. (1969). Minimum Spanning Tree, Algorithm AS12, Applied Statistics, 18, 103-1-4.

12. Sangiovanni-Vincentelli, A., Chen, L., and Chua, L.O. (1977). An efficient heuristic cluster algorithm for tearing large-scale networks. IEEE Transactions on Circuits and Systems, Vol CAS-24, 12, 709-717.

13. Wong, M.A. (1979). Hybrid Clustering, Unpublished Ph.D. Dissertation, Yale University, Department of Statistics.

14. Zahn, C.T. (1971). Graph-theoretic Methods for Detecting and Describing Gestalt Clusters. IEEE Transactions on Computers, Vol. C20, No. 1, 68-86.

DAT

FILM